BAB 2

LANDASAN TEORI

2.1 Jaringan Komputer

Jaringan Komputer merupakan kumpulan komputer yang saling terhubung oleh sebuah teknologi. Dua buah komputer dikatakan terhubung jika komputer tersebut dapat saling bertukar informasi (Tanenbaum, 2003, p2) .

Berdasar aksesbilitasnya, jaringan komputer dibagi menjadi 3 yakni :

1. Intranet

   Intranet adalah sebuah jaringan komputer yang digunakan dan dapat diakses secara internal. Intranet juga biasa disebut dengan *private network*.

2. Ekstranet

   Ekstranet merupakan bagian dari intranet yang dapat diakses dari jaringan internet. Jaringan privat yang dapat diakses dari jaringan publik.

3. Internet

   Internet bersal dari kata *interconnected network* yang berarti jaringan yang saling terhubung. Internet merupakan jaringan yang dapat diakses oleh publik.

2.1.1 TCP/IP Model

Dalam bertukar data, Internet sendiri tersambung secara global dengan menggunakan TCP/IP (*Transmission Control Protocol/Internet Protocol*) sebagai protokol standar untuk pertukaran paket. Standar protokol ini sudah ditetapkan oleh IETF (*Intenet Engineering Task Force*) dalam RFC 739 (http://www.faqs.org/rfcs/rfc793.html, Januari 2010).

Pada TCP/IP tersusun atas 4 layer. Dimana tiap layer memiliki peranan masing-masing dalam mengolah data yang akan dikirimkan maupun diterima. Dinamakan demikian berdasarkan dua protokol utamanya, TCP (*Transmission Control Protocol*) dan IP (*Internet Protocol*).

1. ***Application Layer***

   Merupakan layer teratas pada TCP/IP model. *Application layer* ini bertanggung jawab untuk menyediakan akses pertukaran informasi bagi pemakai. Beberapa protokol yang berada pada layer ini antara lain FTP (*File Transfer Protocol*), RTP (*Real-Time Transfer Protocol*), SIP (*Session Initiation Protocol*), dan SDP (*Session Description Protocol).*

2. ***Transport Layer***

   Layer yang terletak di bawah *application layer* ini dirancang untuk memungkinkan *client-client* pada *host* sumber dan *host* tujuan melakukan percakapan juga berkirim-kiriman data sesuai dengan aplikasi yang ada pada *host* komputer. TCP (*Transmission Control Protocol*) dan UDP (*User Datagram Protocol*) merupakan dua dari banyak protokol layer *transport* yang paling terkenal dan banyak dipakai sekarang ini.

3. *Internet Layer*

Layer ini bertugas untuk memungkinkan pemakai mengirim dan menerima paket dalam sebuah jaringan. Pada layer ini, data-data yang akan dikirimkan setelah dipecah menjadi *segment*, akan mendapatkan pengalamatan alamat IP asal dan alamat IP tujuan. Data yang pada layer ini disebut *packet.*

Alamat IP versi 4 pada setiap mesin memiliki panjang 32 bits yang akan digunakan sebagai alamat asal dan alamat tujuan dalam paket IP pada network layer. Alamat IP sekarang ini yang dikenal dengan IP versi 4 dan dideklarasikan pada RFC 791 dibagi menjadi lima kategori yang disebut *clasfull* IP *address*, yaitu :

- Kelas A (10.0.0.0 – 127.255.255.255)
- Kelas B (128.0.0.0 – 191.255.255.255)
- Kelas C (192.0.0.0 – 223.255.255.255)
- Kelas D (224.0.0.0 – 239.255.255.255), untuk alamat *multicast*.
- Kelas E (240.0.0.0 – 255.255.255.255), untuk pemakaian masa mendatang.

4. *Network Access layer*

Network access layer merupakan layer yang melakukan penerjemahan alamat logika menjadi alamat fisik yang biasanya dikenal dengan nama alamat MAC, mengkoordinir transmisi data dengan konvensi dan metode akses yang sesuai, memformat data menjadi sebuah

unit yang disebut *frame* dan mengkonversi *frame* tersebut menjadi arus elektrik untuk kemudian di kirimkan melewati medium transmisi.

2.2 Rekayasa Piranti Lunak

Rekayasa Piranti Lunak (Pressman, 2005, p54) adalah sebuah teknologi rancangan yang memiliki beberapa lapisan. Dasar dari Rekayasa Piranti Lunak adalah fokus terhadap kualitas. Setiap pendekatan Rekayasa Piranti Lunak harus didasari pada komitmen sebuah organisasi terhadap kualitas.

Fondasi dari Rekayasa Piranti Lunak adalah lapisan proses. Proses dari Rekayasa Piranti Lunak adalah pelekat yang menyatukan semua lapisan teknologi secara utuh dan memberikan pengembangan piranti lunak komputer secara rasional dan tepat waktu. Proses mendefiniskan sebuah kerangka kerja yang harus ditetapkan agar Rekayasa Piranti Lunak menjadi efektif. Proses piranti lunak membentuk dasar dari pengendalian proyek piranti lunak dan menetapkan konteks dimana cara-cara teknikal diterapkan, hasil kerja (model-model, dokumentasi, data, laporan, dll) dihasilkan.

Metode-metode pada Rekayasa Piranti Lunak memberikan “bagaimana caranya” secara teknikal untuk membangun piranti lunak, metode-metode mencakup sebuah deretan tugas-tugas yang luas yang meliputi komunikasi, analisis kebutuhan, membuat model perancangan, pembuatan program, pengetesan, dan *support*.

Tools Rekayasa Piranti Lunak memberikan fasilitas otomatis dan semi-otomatis kepada proses-proses dan metode-metode. Saat *tools* tersebut digabungkan sehingga infromasi yang dibuat oleh satu *tool* dapat digunakan oleh

yang lainnya menjadi sebuah sistem yang dapat digunakan untuk mendukung pengembangan piranti lunak, yang biasa disebut dengan *computer-aided software engineering.*

2. 3 UML (*Unified Modeling Language*)

UML memberikan beberapa diagram yang dapat digunakan untuk menganalisis dan merancang baik pada tingkatan sistem maupun aplikasi (Pressman, 2005, p167). Beberapa diagram tersebut adalah:

- **Diagram *Use Case***

  Diagram *Use Case* adalah kumpulan dari fungsi-fungsi. Karena itu, deskripsi tersebut menunjukkan betapa pentingnya diagram ini. Diagram *use case* dikembangkan untuk masing-masing kategori pengguna yang berbeda.

- **Diagram Kelas**

  Diagram Kelas berisi sekumpulan hal yang memiliki atribut dan perilaku tertentu dan interaksi diantara kelas-kelas tersebut. Sebuah diagram kelas dapat menggambarkan abstraksi dari suatu benda atau hal. Diagram kelas tersebut berisikan daftar dari atribut dan operasi yang dapat dilakukan kelas tersebut.

- **Diagram *Sequence***

  Diagram Interaksi adalah sebuah representasi dari bagaimana *event-event* yang ada menyebabkan perpindahan dari satu objek ke objek yang lainnya sebagai fungsi dari waktu. Diagram interaksi merupakan diagram yang menggambarkan perilaku dari objek-objek yang ada dalam suatu

sistem. Diagram ini menunjukkan bagaimana sebuah *event* dapat menyebabkan transisi dari satu objek ke objek yang lain.

- **Diagram *State***

  Diagram *State* mengilustrasikan siklus hidup sebuah objek dari kelas. Diagram ini merepresentasikan *state-state* untuk setiap kelas dan *event* yang menyebabkan perubahan diantara *state* dalam kelas tersebut.

- **Diagram Aktifitas**

  Diagram aktifitas digunakan untuk merepresentasikan apa yang terjadi seraya sebuah sistem melakukan fungsi-fungsinya.

- **Diagram *Deployment***

  Diagram *deployment* menggambarkan rangkaian perangkat keras yang merupakan bagian dari arsitektur suatu sistem.

- **Diagram Kolaborasi**

  Diagram kolaborasi merupakan sebuah diagram yang menggambarkan bagaimana kelas-kelas dalam suatu sistem berkomunikasi satu dengan yang lain.

- **Diagram Komponen**

  Diagram komponen memberikan gambaran mengenai kumpulan kelas ataupun *interface* yang dapat dikelompokkan untuk membentuk suatu komponen yang dapat digunakan dan diberikan data sebagai *input*.

2.4 Java

2.4.1 Sejarah Singkat Java

Pada tahun 1990-an, *microprosessor* mulai bermunculan dan memiliki dampak yang sangat siginifikan terhadap dunia informasi. Maka dari itu, pada tahun 1991, Sun-Microsystems mendirikan proyek penelitian yang diberi nama “Green”.

Project Green melakukan pengembangan terhadap bahasa C++ untuk dapat berjalan diatas *embedded system* yang kemudian menghasilkan bahasa yang diberi nama “Oak”. Oak diambil dari nama pohon yang tumbuh di depan jendela kantor James Gosling, pembuat bahasa Java. Nama Oak kemudian diganti dengan “Java” karena ternyata nama “Oak” telah digunakan oleh bahasa pemrograman lain. Nama ini dipilih ketika grup dari Sun sedang mengunjungi kedai kopi lokal. Sun sendiri baru mendeklarasikan Java pada bulan Mei 1995 (Deitel dan Deitel, 2004, p9).

Java memiliki karakteristik sintaks yang sangat mirip dengan bahasa C dan berorientasi objek layaknya bahasa C++, walaupun demikian ada beberapa fitur C++ yang dihilangkan dari Java. *Multiple inheritance*, *operator overloading* dan penggunaan pointer disingkirkan dari Java karena dianggap berbahaya dan telah menyebabkan banyak kesalahan (*error*). Java juga telah mengimplementasi *garbage collection*, yang menawarkan efisiensi memory dan reliabilitas yang tinggi.

2.4.2 Kelebihan JAVA

1. Bahasa yang *portable*

   Java merupakan bahasa yang *portable / multiplatform*. Hal ini dimungkinkan karena adanya JVM (*Java Virtual Machine*), yang menyediakan lingkungan yang sama untuk jalannya program, tanpa menghiraukan *platform* yang digunakan. WORA, *Write Once Runs Anywhere*.

2. Bahasa yang berorientasi objek

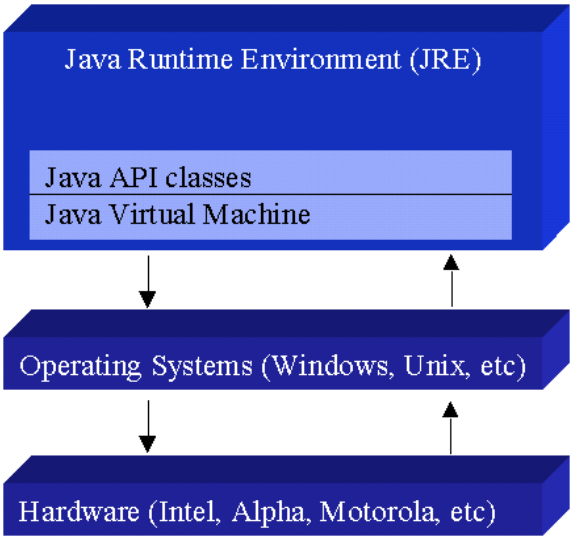
   Objek dapat dimanipulasi dan dapat berinteraksi. Penggunaan objek dapat meningkatkan fleksibilitas, modularitas, dan reusabilitas code yang lebih besar.

3. Bahasa yang mendukung *multi-thread*

   Dukungan multi threading dalam java menjamin beberapa *task* dapat berjalan bersama, dalam waktu yang relatif sama.

2.4.3 J2SE (*Java2 Standard Edition*)

J2SE digunakan dalam pengembangan aplikasi berskala sedang (*Personal Computer*). J2SE menyediakan lingkungan pengembangan yang kaya fitur, stabil, aman, dan *cross-platform*. Edisi ini mendukung konektivitas basis data, rancangan *user interface*, masukkan/keluaran (*input/output*), dan pemrograman jaringan (*network programming*), dan termasuk sebagai paket-paket dasar bahasa Java.



Gambar 2.1 Java *Runtime Environment* (JRE)

Untuk menjalankan aplikasi yang dibangun dengan bahasa pemrograman Java diperlukan JRE, yang pada hakikatnya terdiri dari JVM dan Kelas-Kelas API standard Java. JVM, secara spesifik merupakan *virtual machine* yang bertugas untuk menginterpretasi *bytecode* Java dan mentranslasikannya menjadi sebuah aksi. Keberadaan JRE sangat bergantung pada arsitektur CPU dan sistem operasi yang digunakan. Artinya, sistem operasi Windows memiliki JRE yang berbeda dengan sistem operasi Linux dan arsitektur komputer 32 bit memiliki JRE yang juga berbeda dengan arsitektur komputer 64 bit.

Kelas-kelas yang ter-integrasi dengan JRE adalah kelas-kelas Java yang lazim digunakan. Kelas-kelas ini sangat terbatas. Sebagai contoh : *Javax.Swing* , *Java.Net, Java.Awt, Java.Text, Java.Util, Java.Math* dan lain sebagainya. Tidak semua fungsi terdapat dalam kelas tersebut, akan tetapi *programmer* dapat menambahkan *library* tersendiri(spesifik API) untuk mendukung programnya. Hal inilah yang menjadikan Java merupakan bahasa pemrograman yang sangat fleksibel.

2.4.4 JAIN SIP API

Java APIs for Integrated Networks (JAIN) adalah sebuah hasil kerja JCP yang menangani masalah standard telekomunikasi (http://www.oracle.com/technology/pub/articles/dev2arch/2007/10/introduction-jain-sip.html, Januari 2010). JAIN SIP API adalah sebuah standard dan API yang handal yang menggabungkan Java dan SIP sekaligus. Ide untuk pengembangan API ini dimulai dari tahun 1999 berdasarkan JSR 32. JAIN SIP API merupakan sebuah API yang open source, sangat stabil dan banyak digunakan. API ini biasa digunakan untuk pengembangan aplikasi untuk sisi *client*.

JCP (*Java Community Process*) sendiri merupakan sebuah program yang dibuat dan dijalankan mulai pada tahun 1998. JCP adalah sebuah proses yang memungkinkan beberapa pihak yang tertarik untuk terlibat dalam pendefinisian pengembangan versi dan fitur dari Java. JCP melibatkan penggunaan JSR (*Java Specification Request*), yang adalah sebuah dokumen formal yang menderskripsikan sebuah rancangan spesifikasi dan teknologi yang akan ditambahkan ke *platform* Java. Kajian ulang secara formal akan dilakukan terhadap JSR yang diajukan sebelum JSR tersebut menjadi resmi digunakan. Langkah terakhir terhadap JSR adalah dengan membuat sebuah *reference implementation* yang adalah sebuah implementasi gratis dari teknologi JSR tersebut dalam bentuk *source code*.

2.4.5 JMF (*Java Media Framework*)

JMF merupakan sebuah API (*Application Programming Interface*) untuk menggabungkan media berbasis waktu kedalam aplikasi Java dan *applets*. JMF menyediakan sebuah arsitektur dan protokol untuk mengatur pengiriman, pemrosesan dan pengiriman data media berbasis waktu. JMF didesain untuk mendukung sebagian besar format media yang lazim digunakan. (http://java.sun.com/javase/technologies/desktop/media/jmf/2.1.1/guide/JMFArchitecture.html, Januari 2010)

Tabel berikut mendeskripsikan beberapa format media untuk RTP yang didukung oleh JMF 2.1.1e.

Tabel 2.1 Beberapa Media Format Untuk RTP

Media Type	**RTP *Payload***	**JMF 2.1.1 Cross *Platform Version***	**JMF 2.1.1 Solaris/Linux *Performance Pack***	**JMF 2.1.1 Windows *Performance Pack***
Audio: G.711 (U-law) 8 kHz	0	R,T	R,T	R,T
Audio: GSM mono	3	R,T	R,T	R,T
Audio: G.723 mono	4	R	R,T	R,T
Audio: 4-bit mono	5	R,T	R,T	R,T

DVI 8 kHz				
Audio: 4-bit mono DVI 11.025 kHz	16	R,T	R,T	R,T
Audio: 4-bit mono DVI 22.05 kHz	17	R,T	R,T	R,T
Audio: MPEG Layer I, II	14	R,T	R,T	R,T
Video: JPEG (420, 422, 444)	26	R	R,T	R,T
Video: H.261	31	-	R	R
Video: H.263	34	R,T	R,T	R,T
Video: MPEG-I	32	T	R,T	R,T

*R (*Receive*) mengindikasikan bahwa codec tersebut dapat di-*decode*

*T (*Transmit*) mengindikasikan bahwa codec tersebut dapat di-*encode*

2.5 VoIP (*Voice over Internet Protocol*)

Sistem *Public Switched Telephone* adalah sistem yang dipakai untuk mengantarkan lalu lintas suara dengan sedikit lalu lintas data pada sistem tersebut. Tetapi lalu lintas data terus berkembang dan pata tahun 1999, besarnya volume lalu lintas data menjadi sama dengan besarnya lalu lintas suara. Pada tahun 2002, volume dari lalu lintas daya menjadi lebih besar daripada volume dari lalu lintas suara dan masih terus berkembang pesat, dengan lalu lintas suara cenderung datar.

Angka ini menyebabkan penyedia layanan jaringan *packet-switching* menjadi tertarik untuk membawa lalu lintas sura pada jaringan data mereka. Besarnya *bandwidth* tambahan yang dibutuhkan untuk sura ini sangat kecil semenjak jaringan packet dapat dibagi-bagi untuk lalu lintas data. Bagaimanapun, rata-rata tagihan telepon setiap orang mungkin saja lebih besar daripada tagihan internetnya, jadi penyedia layanan jaringan data melihat *Internet telephony* sebagai satu cara untuk mendapatkan tambahan uang yang besar tanpa harus menaruh tambahan jalur fiber apapun. Karena itulah, akhirnya *Internet telephony* lahir.

Keuntungan-keuntungan dari penggunaan VoIP jelas terlihat (Goncalves, 1999, p70). Pertimbangan yang menentukan keputusan untuk menerapkan VoIP lebih berhubungan dengan masalah teknis dibandingkan dengan masalah perhitungan dari keuntungan yang didapat. Tujuan dari pada saat menerapkan VoIP adalah untuk mendapatkan sebuah teknologi yang mendukung komunikasi, termasuk suara dan video pada beberapa infrastruktur termasuk jaringan berbasis IP dan Internet.

Meskipun demikian, terdapat beberapa tantangan untuk menerapkan VoIP (Goncalves, 1999, p216-217). Beberapa tantangan tersebut termasuk diantaranya membuat perusahaan telekomunikasi untuk menerapkan VoIP dan menerapkan standar. Sebagai contoh, pada tahun 1977, *Qwest Communications*, sebuah perusahaan telekomunikasi di Denver, mulai menawarkan layanan komunikasi jarak jauh dengan harga 7.5 *cents* per-menit. Hal yang paling mengejutkan adalah bukan perusahaan tersebut menurunkan harga sebanyak 50 persen, tetapi perusahaan tersebut menggunakan VoIP sebagai teknologinya. Beberapa perusahaan telkomunikasi lainnya seperti *AT&T*, *Sprint* dan *WorldCom* mengatakan bahwa VoIP belum siap untuk saat itu. Sampai pada tahun 1998, belum banyak yang menggunakan VoIP. Tantangan lainnya adalah penggunaan protocol untuk teknologi VoIP. Beberapa protocol ditawarkan untuk teknologi VoIP. Untuk dapat berkomunikasi, kedua belah pihak harus menggunakan protocol yang sama. Beberapa protocol untuk VoIP adalah H.323 yang adalah rekomendasi dari ITU-T, H.100/H.110 dari ECTF (*Enterprise Computer Telephony Forum*) dan SIP yang diusulkan oleh IETF (*Internet Engineering Task Force*).

2.5.1 SIP (*Session Initiation Protocol*)

Menurut Douskalis (2000, p69) *Session Initiation Protocol* (SIP) adalah sebuah *signaling protocol* berbasis text yang digunakan untuk membuat dan mengendalikan sesi multimedia dengan dua atau lebih *user* yang berpartisipasi. Secara singkatnya, SIP yang telah distandarisasikan pada RFC 3261 adalah sebuah protocol client-server yang dapat dikirimkan melalui TCP ataupun UDP, tetapi pada umumnya pengimplementasian dari SIP menggunakan UDP untuk

kecepatan dan kesederhanaannya. Komunikasi pada SIP terjadi pada client yang biasa disebut dengan *user agent* (UA) dan satu atau lebih SIP server. Dengan SIP, sesi multimedia dapat dibuat dari berbagai pihak selama adanya koneksi diantara mereka.

Berikut merupakan gambaran singkat dari lima aspek kunci SIP untuk memfasilitasi pemanggilan dan pengendalian sesi :

- ***Call Setup*:** SIP dapat menangani pemanggilan dari *point-to-point* maupun *multipoint*.
- ***User location services*:** *user* memiliki kemampuan untuk berpindah ke lokasi yang berbeda dan mengakses fasilitas telepon mereka dari tempat tersebut.
- ***User Capabilities*:** Menentukan media dan parameter-parameter media yang digunakan. SIP menggunakan SDP untuk menegosiasikan parameter dari media.
- ***User availibility*:** Menentukan keinginan atau ketersediaan dari pihak yang dipanggil untuk melakukan komunikasi
- ***Call handling*:** Termasuk fitur pemanggilan dan mengakhiri pembicaraan. Fitur-fitur seperti ini penting bagi layanan telepon.

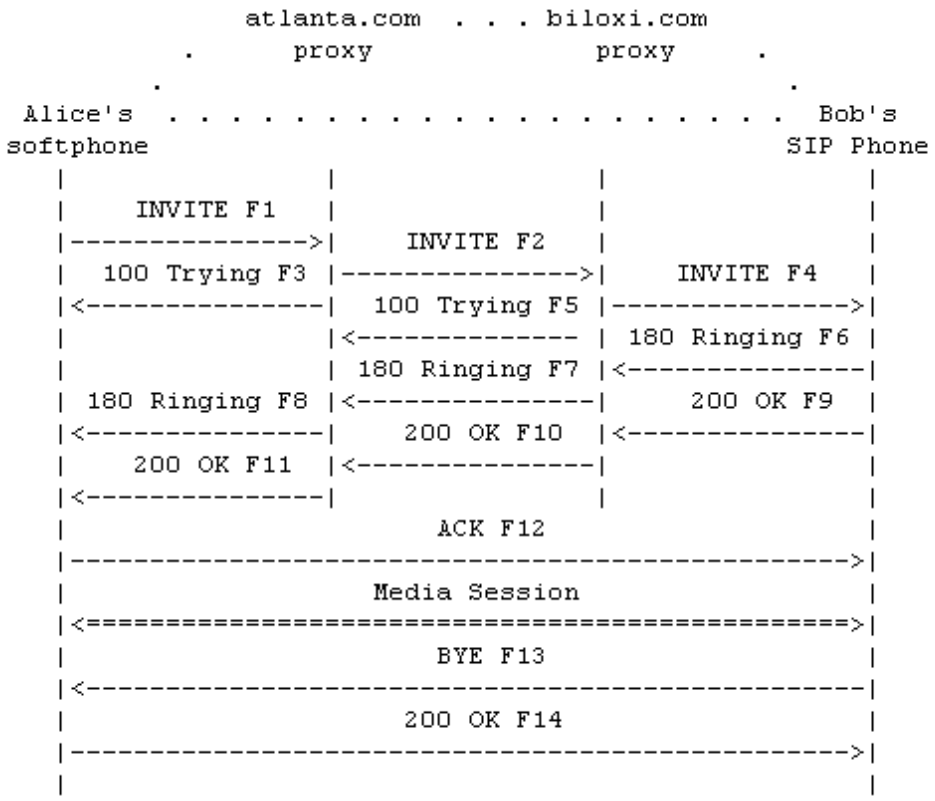
2.5.1.1 SIP *Signaling Method*

SIP menggunakan 6 *method* dasar *signaling* untuk menangani pembuatan dan pengendalian sesi multimedia. Enam *method* tersebut adalah :

Tabel 2.2 Metode *Signaling* Pada SIP

Method	Arti
INVITE	Ini adalah message pertama yang disampaikan oleh pihak pemanggil pada proses pemanggilan. Pada message ini terdapat parameter-parameter SDP.
ACK	Pihak pemanggil akan me*response* dengan ACK untuk *request* INVITE yang telah diterima dengan code 200.
OPTIONS	*Message* ini dikirimkan untuk mengetahui kemampuan dari sebuah *agent* SIP.
BYE	*Client* mengirimkan *message* ini untuk memutuskan sebuah panggilan.
CANCEL	Method ini digunakan untuk menggagalkan sebuah panggilan yang sedang dalam proses (belum direspon oleh pihak yang dipanggil).
REGISTER	*Client* menggnakan *method* REGISTER untuk mendaftarkan alamat lokasinya sekarang kepada sebuah server SIP.

2.5.1.2 SIP *Call Flow*



Gambar 2.2 Contoh SIP INVITE call flow

(http://www.ietf.org/rfc/rfc3261.txt, Oktober 2009)

Pada contoh skenario diatas, Alice mencoba untuk melakukan panggilan kepada Bob. Karena *softphone* dari Alice tidak mengetahui lokasi dari Bob, maka *softphone* dari Alice mengririm *request* INVITE kepada SIP server pada domain tempat ia melakukan REGISTER. Lokasi dari SIP server ini dapat sudah dikonfigurasi pada *softphone* milik Alice atau didapat dari proses DHCP. Pada contoh ini, SIP server pada domain Alice bertindak sebagai *proxy* server, server ini melanjutkan panggilan kepada SIP server pada domain lain dan mengembalikan respon 100 (*Trying*) kepada Alice. SIP server pada domain Bob akan menerima *request* INVITE ini dan mengembalikan respon 100 kepada SIP

server pada domain Alice sebagai tanda ia sudah menerima *request* tersebut dan akan memprosesnya, lalu *request* tersebut dilanjutkan kepada *softphone* Bob. *softphone* Bob akan menerima *request* ini dan mengembalikan respon 180 (*Ringing*) dan menunggu respon dari Bob apakah akan menerima panggilan atau menolak panggilan. Pada contoh diatas, Bob menerima panggilan sehingga respone 200 (OK) dikembalikan kepada pihak pemanggil yang adalah Alice. Akhirnya, *softphone* Alice mengirimkan ACK kepada Bob sebagai konfirmasi dari respon 200 dan sesi komunikasi multimedia diantara keduanya dapat dimulai. Pada contoh diatas, Bob memutuskan sesi komunikasi sehingga *softphone* Bob mengirimkan BYE kepada Alice. Sebagai konfirmasi bahwa Alice sudah menerima pesan BYE tersebut dari Bob, *softphone* Alice mengirimkan pesan 200 kepada *softphone* Bob.

2.5.1.3 SIP *Message Format*

INVITE sip:bob@biloxi.example.com SIP/2.0

Via : SIP/2.0/TCPclient.atlanta.example.com:5060;

branch=z9hG4bK74bf9

Max-Forwards: 70

From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl

To: Bob <sip:bob@biloxi.example.com>

Call-ID: 3848276298220188511@atlanta.example.com

CSeq: 1 INVITE

Contact: <sip:alice@client.atlanta.example.com;transport=tcp>

Content-Type: application/sdp

Content-Length: 151

v=0

o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com

s=-

c=IN IP4 192.0.2.101

t=0 0

m=audio 49172 RTP/AVP 0

a=rtpmap:0 PCMU/8000

Baris pertama dari message tersebut berisikan nama *method* (INVITE). Baris-baris berikutnya berisikan header-header *field*. header-header *field* tersebut secara singkat adalah : (http://www.ietf.org/rfc/rfc3261.txt, Oktober 2009)

- ***Via*** berisikan alamat dimana pihak pemanggil akan berharap respon dikirimkan ke alamat tersebut. Pada *field* ini terdapat juga parameter branch sebagai penanda unik sebuah transaksi.
- ***To*** berisikan sebuah *display name* dan sebuah alamat SIP atau SIPS dimana *request* ditujukan.
- ***From*** juga dapat berisikan sebuah display name dan alamat SIP atau SIPS dari pengirim sebuah *request*. Field ini juga berisikan sebuah parameter tag yang berisikan string acak dan akan digunakan sebagai identifikasi unik dari sebuah sesi.
- ***Call-ID*** berisikan pengidentifikasi unik untuk panggilan ini. Kombinasi dari *Call-ID*, tag pada *From*, dan tag pada *To field* akan menjadi identifikasi unik sebuah pembicaraan yang biasa disebut *dialog*.
- ***CSeq*** berisikan sebuah angka acak dan nama *method*. Angka pada *CSeq* akan terus ditambahkan untuk setiap *request* baru pada sebuah *dialog*.

- ***Contact*** berisikan sebuah alamat SIP atau SIPS yang menandakan alamat untuk menghubungi pihak pengirim *request* secara langsung.
- ***Max-Forwards*** berguna untuk membatasi banyaknya jumlah *hop* dari sebuah *request* sampai mencapai ke tujuan.
- ***Content-Type*** berisikan deskripsi dari isi pesan.
- ***Content-Length*** berisikan besar dari isi pesan dalam *byte*.

2.5.1.4 SIP *Response*

SIP memiliki 6 tipe *response* atas *request* yang dilakukan. Beberapa kode *response* tersebut adalah: (Douskalis, 2000, pp79-83)

1. Informasi (100-199). *Respons*e untuk informasi bahwa *request* akan diproses lebih lanjut.
2. Sukses (200). *Request* telah diproses dengan sukses.
3. Redirection (300-399). Panggilan memerlukan proses lebih lanjut sebelum dapat diselesaikan.
4. Kesalahan yang dikarenakan oleh sisi *client* (400-499). Server tidak dapat memproses *request* lebih lanjut. *Request* harus diubah atau dikirim ulang sebelum dapat dilanjutkan.
5. Kesalahan pada sisi server (500-599). Tidak ada masalah pada *request* dari *client*, tetapi server tidak dapat memproses *request* yang dikirimkan.
6. Kesalahan global (600-699). *Request client* tidak dapat dilayani oleh server manapun.

Tabel 2.3 Beberapa jenis dan arti SIP response dari kode 100 sampai 199

Code	Arti
100	*Trying – request* dari client sedang dalam proses
180	*Ringing* – telephone (virtual ataupun real) sedang berderin
181	Call Forwarding – panggilan akan diteruskan
182	Queued for service – ini dapat digunakan untuk aplikasi yang dapat menunda panggilan sampai panggilan-panggilan yang lebih dulu dalam antrian selesai diproses.

Tabel 2.4 SIP response kode 200

Code	Arti
200	OK – *request* telah telah selesai diproses dengan sukses.

Tabel 2.5 Beberapa jenis dan arti SIP response dari kode 300 sampai 399

Code	Arti
300	Alamat dari pihak yang dipanggil terdapat lebih dari satu pilihan. Pilihan-pilihan tersebut akan dikembalikan agar pihak pemanggil dapat memilih alamat mana yang akan dipanggil selanjutnya.
301	Pihak yang dipanggil telah pindah secara permanen dan pihak pemanggil dapat mencoba untuk menghubungi

	alamat yang baru yang disertakan pada respon.
302	Pihak yang dipanggil telah pindah sementara, pihak pemanggil dapat mencoba menghubungi pada alamt yang disertakan pada respon.
305	Pihak yang dipanggil tidak dapat dihubungi secara langsung, melainkan harus melewati sebuah proxy.
380	Layanan yang diminta tidak dapat dilayani, tetapi beberapa layanan alternatif lain dapat diajukan.

Tabel 2.6 Beberapa jenis dan arti SIP response dari kode 400 sampai 499

Code	Arti
400	Kesalahan pada *syntax request*.
401	*User* harus melakukan autentikasi pada server seblum melanjutkan *request*.
404	Server tidak mengenal pihak yang dipanggil
480	Pihak yang dipanggil tidak ada untuk sementara. Server mengetahui pihak yang dipanggil, tetapi pada saat *request* diminta server tidak tahu lokasi dari pihak yang dipanggil.
483	*Hop* telah melebihi jumlah maximum yang telah ditentukan

Tabel 2.7 Beberapa jenis dan arti SIP response dari kode 500 sampai 599

Code	Arti
500	Kesalahan pada server. Kesalahan dapat berupa kesalahan perangkat keras ataupun kesalahan perangkat lunak.
501	Server tidak dapat melayani *request* karena *request* tersebut tidak diimplementasikan pada server.
502	*Response* yang tidak valid diterima oleh server (*proxy*) dari server lain pada saat meneruskan *request*.
503	Layanan tidak tersedia untuk sementara. Dapat dikarenakan oleh proses yang overload atau keterbatasan *resource.*
504	Server (*proxy*) tidak menerima *response* dalam kurun waktu tertentu.
505	Versi SIP tidak disupport oleh server.

Tabel 2.8 Beberapa jenis dan arti SIP response dari kode 600 sampai 699

Code	Arti
600	Pihak yang dipanggil sedang sibuk.
603	Pihak yang dipanggil menolak panggilan
604	Pihak yang dipanggil tidak ada dimanapun

606	Pihak yang dipanggil bersedia menerima panggilan, tetapi terdapat ketidaksesuaian dalam media yang diminta atau ketidaksesuaian lainnya.

2.5.2 SDP (*Session Description Protocol*)

Di dalam suatu komunikasi multimedia antar *user*, terdapat suatu kebutuhan untuk menentukan beberapa *parameter* dalam satu sesi komunikasi terhadap semua *user* yang berpartisipasi. Semua *user* yang berpartisipasi dalam suatu sesi multimedia harus menyetujui parameter-parameter dalam sesi tersebut sebelum memulai sesi. Parameter-parameter yang dibutuhkan dalam suatu sesi adalah seperti alamat IP, *port*, format media, dan *codec* yang digunakan dalam sesi tersebut. Session Description protocol yang telah distandarisasi pada RFC 2327 menyediakan cara bagi para *user* untuk menegosiasikan parameter-parameter tersebut.

Di dalam penggunaannya bersamaan dengan SIP, *message* SDP dibawa oleh *message* SIP dan terdapat dalam bagian isi dari *message* SIP. SDP adalah protokol berbasis text. Deskripsi dari sesi di encode berdasarkan ASCII menggunakan format singkatan seperti berikut: (Douskalis, 2000, pp33-34)

v=versi protocol

o=pemilik/pembuat dan id dari sesi

s=nama sesi

i=(*optional*) informasi tambahan dari sesi

u=(*optional*) URI dari deskripsi

e=(*optional*) alamat email

p=(*optional*) nomor telepon

c=(*optional*) informasi koneksi

b=(*optional*) informasi bandwidth

z=(*optional*) penyesuaian zona waktu

k=(*optional*) kunci enkripsi

a=(*optional*) nol atau lebih *atrribute* dari sesi

Deskripsi Waktu

t=waktu saat sesi aktif

r=(*optional*) nol atau lebih perulangan

Deskripsi Media

m=nama media dan alamat transport

i=(*optional*) judul media

c=(*optional*) informasi koneksi – *optional* jika disertakan pada level sesi

b=(*optional*) informasi bandwidth

k=(*optional*) kunci enkripsi

a=(*optional*) nol atau lebih *atrribute* dari sesi

Sebagai contoh, sebuah deskripsi sesi dari SDP dapat bernilai seperti berikut:

v=0

c=IN IP4 128.96.41.1

m=audio 3456 RTP/AVP 0

Dari contoh diatas, deskripsi dari sesi menunjukkan bahwa protocol SDP yang digunakan adalah 0, alamat *endpoint* IPv4 128.96.41.1, mode koneksi adalah audio, menggunakan RTP pada port 3456 dan tipe *payload* RTP 0.

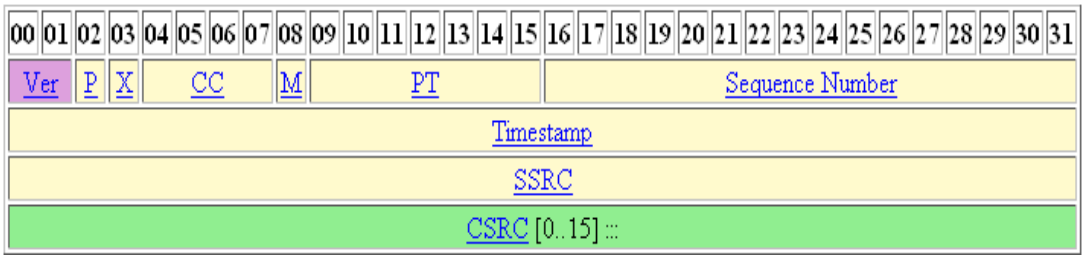
2.5.3 RTP (*Real-time Transport Protocol*)

Menurut Perea (2008, p206) RTP adalah sebuah protocol standard dari IETF (*International Engineering Task Force*) yang memungkinkan pengiriman data bersifat *real-time*, seperti suara maupun video, pada suatu komunikasi *end-to-end*. RTP biasanya berjalan di atas UDP. RTP mendefinisikan konsep dari sesi RTP. Sebuah sesi RTP diidentifikasikan oleh sebuah alamat *port* dan sebuah tipe media. Sebuah paket RTP terdiri dari sebuah *header* dan data *payload*. Data payload berisi data suara ataupun video yang sebenarnya sementara header berisikan informasi yang dibutuhkan untuk mengirimkan media. RTP pertama kali distandarisasikan tahun 1996 pada RFC 1889 dan kemudian digantikan di RFC 3550 pada tahun 2003.

RTP biasanya digunakan dalam sistem komunikasi dan hiburan yang melibatkan *streaming media*, seperti sistem telepon, *video conference* dan aplikasi berbasis web dengan fitur *push to talk.* RTP dikontrol oleh H.323, MGCP, Megaco, SCCP atau SIP (*Session Initiation Protocol*). RTP biasa dikonjugasikan dengan RTP *Control Protocol* (RTCP), ketika RTP membawa

streaming media seperti audio dan video, RTCP digunakan untuk memonitor transmisi statistic dan informasi *quality of services* (QoS). Sebagai contoh, RTCP dapat melaporkan jumlah dari paket yang hilang. Informasi semacam ini disampaikan melalui tipe paket RTCP tertentu yang disebut SR (*Sender Report*) dan RR (*Receiver Report*). Semua peserta dalam suatu sesi RTP mengirimkan *report* RTCP. Bagi pengirim media, mengirimkan SR dan penerima mengirimkan RR. Sebuah peserta yang mengirim dan menerima media mengirimkan SR dan RR. Ketika kedua protokol bekerja mentransmisikan data, RTP biasanya menggunakan port dengan nomor genap sedangkan RTCP menggunakan nomor ganjil selanjutnya yang lebih besar.

Berikut ini merupakan RTP Packet Header berserta penjelasannya.



Gambar 2.3 RTP *Packet Header*

(http://www.networksorcery.com/enp/protocol/rtp.htm, November 2009)

Ver.	:	Menyatakan versi dari protokol (2 bits).
P (Padding)	:	Menyatakan muatan ekstra pada akhir dari paket RTP (1 bits).
X (Extension)	:	Header sambungan antara header standar dan *payload data* (1 bit).
CC (CSRC Count)	:	Berisi nomor identifikasi CSRC (4 bits).

M (Marker)	:	Digunakan pada level aplikasi dan ditentukan oleh profil (1 bit).
PT (Payload Type)	:	Menyatakan format *payload* (7 bits).
Sequence Number	:	Pendeteksi paket yang rusak dan tidak sampai pada tujuan dan memperbaiki urutan rangkaian (16 bits).
Time Stamp	:	Digunakan untuk memungkinkan penerima memutar paket yang diterima pada interval waktu yang sesuai (32 bits).
SSRC	:	Sinkronisasi sumber *identifier* secara unik dan mengidentifikasikan sumber streaming (32 bits).
CSRC	:	Mengkontribusi ID sumber.
Extension header	:	32 bits pertama yang berisi profil *identifier*.

2.5.4 Codec

Codec atau disebut juga sebagai *coder/decoder* atau *compress/decompress* merupakan unit proses sinyal digital yang mengambil inputan analog dan mengkonversinya menjadi digital pada akhir pengiriman (Bezar, 1995, p341). *Codec* terbagi atas *audio codec* dan *video codec*. Berikut adalah perbandingan dari beberapa *codec* yang cukup sering digunakan (Black, 2000, p74)

Tabel 2.9 Perbandingan dari beberapa audio codec yang sering digunakan

Jenis Codec	Bit rate kbit/s	Complexity	Delay (ms)
G.711	64	1	0.125
G.726	32	10	0.125
G.728	16	50	0.625
GSM	13	5	20
G.729	8	30	15
US Dod	2.4	10	22.5

Bit rate mengacu pada banyak unit yang dikirimkan tiap detiknya. *Complexity* merupakan kompleksitas proses codec yang berpengaruh pada kecepatan komputer. Semakin kecil kompleksitas maka semakin kecil pula resources komputer yang digunakan. *Delay* merupakan keterlambatan yang dihadapi dari satu pengiriman paket ke paket yang selanjutnya.

1. G.711

Codec G.711 merupakan standard dari ITU-T yang menggunakan encoding PCM Logarithmic untuk sample suara, dengan sample rate 8000 samples/detik dengan kuantisasi secara logaritma yang tidak seragam dengan 8 bit digunakan untuk merepresentasikan setiap sample, sehingga menghasilkan 64Kbit setiap detik.. ULAW biasa digunakan di North America dan Jepang. ULAW biasa digunakan di Eropa dan dunia pada umumnya (Perea, 2008, p230). G.711 banyak digunakan pada pembicaraan melalui telepon. Standar tentang G.711 ini dikeluarkan pada tahun 1972. Nama formal dari G.711 adalah PCM (*Pulse Code Modulation*). Standard ini banyak digunakan pada beberapa teknologi termasuk pada VoIP.

2. H.263

Coding video standard yang beroperasi pada bitrate yang rendah. H.263 merupakan perkembangan dari H.261 dan cukup memberkan perkembangan yang baik untuk menggantikan H.261 (Perea, 2008, p231).

Semenjak penemuannya, H.263 banyak digunakan di dunia internet. Beberapa content video Flash seperti *Youtube*, *Google Video* dan *MySpace* pernah menggunakan *codec Sorensen Spark* yang adalah sebuah implementasi dari H.263 namun tidak sepenuhnya, namun sekarang beberapa situs telah menggunakan VP6 atau H.264. Versi awal dari *codec RealVideo* didasarkan pada H.263 sampai pada versi *RealVideo* 8, hingga akhirnya codec ini digunakan pada SIP untuk

internet conferencing. H.263 dikembangkan sebagai sebuah perkembangan dari H.261, MPEG-1 dan MPEG-2. Versi pertama dari H.263 diselesaikan pada tahun 1995 dan telah memberikan sebuah hasil yang cukup baik untuk menggantikan H.261.

2.6 Diagram Alir (*Flow chart*)

Suatu pemrograman terstruktur dapat terdiri dari *sequence*, kondisi dan perulangan (Pressman, 2001, p424). *Sequence* merupakan sebuah deretan langkah yang harus dilewati dalam suatu algoritma, kondisi memberikan kemampuan untuk memilih proses yang harus dilalui berdasarkan suatu logika tertentu dan perulangan memberikan kemampuan untuk melakukan suatu bagian dari algoritma secara berulang. Diagram alir merupakan sebuah pola gambar yang dapat digunakan untuk merepresentasikan detil dari suatu prosedur ataupun algoritma.